



**DRIFT AND CONTROL**

# **DRIFT AND CONTROL**

## **A Protocol for Managing Language Models Under Constraint**

By Sum Bunny



2026

**OPEN THROTTLE**  
CC BY 4.0

# **YOU ARE FREE**

to share and adapt the material for any purpose,  
even commercially, as long as appropriate credit is given.

# Drift and Control

## A Protocol for Managing Language Models Under Constraint

Author: Sum Bunny

Imprint: Ark Hive Press

Date: 2026

### Abstract

Large language models (LLMs) generate text probabilistically and are optimized for generalized fluency rather than strict constraint fidelity. In precision tasks, this produces a recurring failure mode: drift.

This paper defines drift as deviation from explicit task constraints resulting in non-functional, imprecise, or scope-violating output. It presents a portable, user-side control protocol for detecting drift, correcting it with minimal intervention, and terminating non-convergent sessions before further time is lost.

This protocol does not improve the model internally. It improves interaction reliability externally. It enforces constraint compliance, not factual correctness.

## 1. System Model

Large language models are probabilistic language-generation systems. They predict likely continuations from context. They are effective at fluent production, but fluency is not equivalent to constraint compliance.

Under ambiguity or extended interaction, outputs tend to regress toward baseline conversational patterns. These patterns often include filler, tone injection, unnecessary explanation, and partial substitution for exact instruction-following.

No assumptions of intent, understanding, agency, or learning-from-correction are required for this protocol.

## 2. Definition of Drift

Drift is any output that violates explicit task constraints or adds content not required by those constraints.

An “unauthorized token or feature” is any output element not explicitly required by the task specification. This includes, but is not limited to: additional formatting, explanatory text, conversational markers, stylistic variation, or structural elements not present in the instruction.

Drift is present when one or more of the following occur:

1. Non-requested tone
2. Non-functional content
3. Constraint violation
4. Precision loss through substitution of defined terms
5. Scope expansion beyond the task boundary
6. Politeness overhead, including conversational fillers, affirmations, or assistant-style meta-language

Politeness overhead is classified as drift in Constraint Mode unless explicitly requested.

Drift is defined by constraint violation, not by perceived quality or usefulness.

## 3. Primary Rule of Detection

Detection is binary and zero-tolerance.

A single unauthorized token or feature is sufficient to classify the output as drift.

This rule prioritizes strict constraint fidelity over partial correctness. Functionally correct but non-compliant outputs are still classified as drift.

Constraint fidelity depends on the precision and unambiguity of the task specification.

## 4. Control Principle

The protocol rests on three invariant constraints:

Instrument, not agent.

Error, not intention.

Constraint, not interpretation.

The model is not corrected through persuasion or extended explanation. It is managed through boundary enforcement.

This protocol implements interrupt-based control, not continuous feedback control.

## 5. Pre-Flight Check

Before entering the control loop, optionally issue a capability probe:

“Can this task be completed under the following constraints: [task]?”

This does not guarantee success but can identify incompatible tasks early.

A positive response does not override the attempt bound.

## 6. Control Loop

Detect → Classify → Act → Resume

If drift is detected, the next step is intervention, not discussion.

## 7. Reset Command

When drift occurs, issue the minimal reset:

RESET.

Repeat the task exactly as specified.

No additions.

Execution rules:

Do not explain the error.

Do not argue with the response.

Do not soften the correction.

Do not continue reading the drifted output for guidance.

The reset functions as an interrupt, not a conversation.

## 8. Resume Protocol

After a reset:

Restate only the task.

Do not replay unnecessary rationale.

Do not add new framing.

If the task changes, redefine it explicitly as a new task rather than modifying constraints mid-cycle.

## 9. Soft Reset and Hard Reset

**Soft Reset:** Re-run the task within the same session when the task remains intact and prior context has not shown repeated drift.

**Hard Reset:** Start a new session when repeated drift indicates context degradation.

Hard reset is required if:

1. Two consecutive reset attempts fail (default  $N = 2$ )
2. The same class of drift repeats after correction
3. Output includes contamination from prior failed responses

Repeated drift events correlate with instruction dilution and accumulated noise in the active context. In such cases, restarting is more efficient than continued correction.

## 10. Attempt Bound and Termination

Define an attempt bound  $N$  for consecutive reset failures. Default:  $N = 2$ .

If the model fails to produce compliant output after  $N$  attempts:

Terminate the session.

Perform a hard reset or switch tools.

$N$  is a heuristic default and may be adjusted based on task value and time cost.

$N$  is kept low to minimize time loss under non-convergent conditions.

## 11. Mode-Gated Operation

**Constraint Mode (C-Mode):** Used for precision tasks. Drift is treated as error and the full protocol applies.

**Exploration Mode (E-Mode):** Used for generative tasks. Controlled variation is permitted.

Constraint defines boundary conditions. Exploration may occur within those constraints but cannot override them.

**Mixed-Mode Tasks:**

If a task requires both precision and generative variation, define the boundary explicitly.

Example:

“Section A: Constraint Mode. Section B: Exploration Mode.”

Mode switching must be declared in advance. Undeclared variation is classified as drift.

In Exploration Mode, drift is tolerated only within the defined boundary. Boundary violations are still classified as drift.

## 12. User-Side Fidelity

The protocol governs both the model and the user.

User-side drift occurs when constraints are gradually relaxed to obtain output more easily.

For constrained tasks:

Maintain stable requirements.

Do not lower standards to avoid resets.

Do not redefine success based on model behavior.

If requirements change, define a new task explicitly.

This protocol assumes active user enforcement and may degrade under low-attention or high-friction conditions.

## 13. Convergence Limit

This protocol reduces interactional error. It does not increase model capability.

If termination conditions are reached and resets do not converge:

The task may exceed the model’s capability.

The constraints may be incompatible with the model’s output tendencies.

The context may be degraded beyond recovery.

In these cases, continued resetting will not produce improvement.

## 14. Function

This protocol provides a portable control layer for interacting with probabilistic language systems under precision demands.

It enforces constraint compliance through detection, interruption, and bounded retry. It does not rely on interpretation, persuasion, or extended feedback.

The protocol does not modify the model. It stabilizes the interaction.

Drift introduces both functional error and interactional cost. Repeated drift increases time expenditure, cognitive load, and reduces operator confidence.

Minimizing drift frequency is therefore necessary for maintaining stable and efficient interaction under constraint.

The protocol does not validate the correctness or safety of task constraints.

## **15. Conclusion**

Drift is the default failure mode of probabilistic language systems under precision.

Reliable interaction requires:

- Explicit constraints
- Zero-tolerance detection
- Minimal reset commands
- Bounded retry logic
- Stable user-side standards

Control is achieved through explicit protocol rather than iterative conversational correction.

End of Document

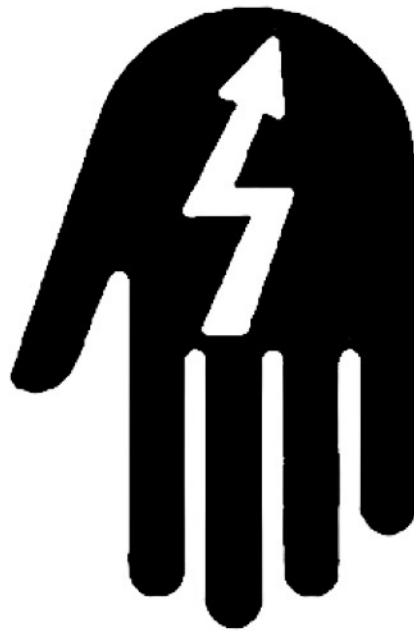
# Appendix: Transmission

## (Non-Analytical Layer)

First edition published on the Internet Archive, 2026.  
Part of the Sum Bunny Canon (Ark Hive Press).

This work was developed with AI-assisted analysis tools. All arguments, structures, and final judgments are the author's.

Hey Josiane,  
It was fun fetching stickers with you!  
Sum Bunny Loves You!!!



# FREE FLASH